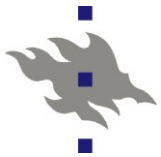
BitTorrent

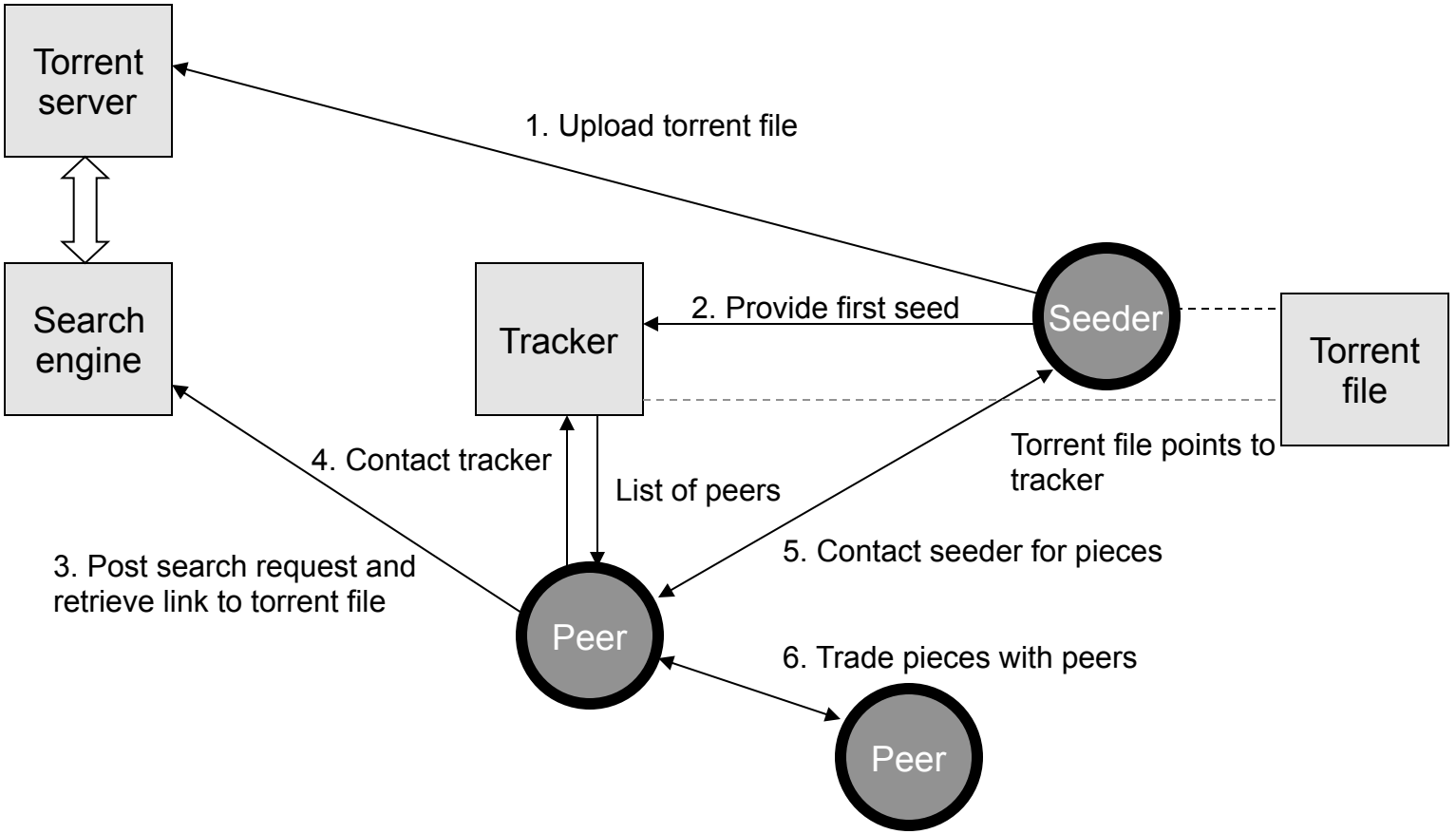
BitTorrent is based on the notion of a **torrent**, which is a smallish file that contains metadata about a host, the tracker, that coordinates the file distribution and files that are shared

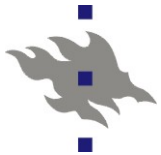
A peer that wishes to make data available must first find a tracker for the data, create a torrent, and then distribute the torrent file. Other peers can then using information contained in the torrent file assist each other in downloading the file

The download is coordinated by the tracker. In BitTorrent terminology, peers that provide a complete file with all of its pieces are called **seeders**



BitTorrent: Downloading Files

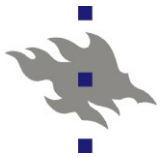




Difference to HTTP

A BitTorrent file download differs from an HTTP request in the following ways:

- BitTorrent uses multiple parallel connections to improve download rates, whereas Web browsers typically use a single TCP Socket to transfer HTTP requests and responses
- BitTorrent is peer-assisted whereas HTTP request is strictly client-server
- BitTorrent uses the random or rarest-first mechanisms to ensure data availability, whereas HTTP is incremental



A solution to the broadcasting problem

BitTorrent attempts to solve the broadcasting problem, which has the goal of disseminating M messages in a population of N nodes in the shortest time

In an environment in which the nodes have bidirectional communications and the same bandwidth, the lower bound on download time (rounds) is given by $M + \log_2 N$, the unit is the time it takes for two nodes to exchange a message

This problem can be solved optimally with a centralized scheduler; however, BitTorrent lacks this centralized component and furthermore it does not have a completely connected graph as well

BitTorrent therefore has a heuristic approach to solving this problem that works very well in practice

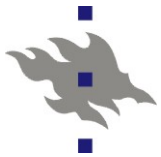
Lower Bound

Assume bidirectional communications and the same bandwidth

The lower bound on download time (rounds) is given by M + $\log_2$ N, the unit is the time it takes for two nodes to exchange a message

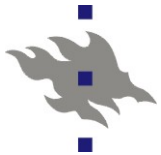
Proof: stat.haifa.ac.il/~gweiss/publications/p2pjos.pdf

Idea: in the first phase one client has the messages, and in the next phase $\log_2$ N rounds are needed to inform the N-1 clients. The log comes from the P2P behaviour in which the clients utilize parallel data transfers to propagate the messages



Characteristics of the BitTorrent protocol I/II

- **Peer selection** is about selecting peers who are willing to share files back to the current peer
  - **Tit for tat** in peer selection based on download-speed.
  - The mechanism uses a **choking/unchoking** mechanism to control peer selection. The goal is to get good TCP performance and mitigate free riders
- **Optimistic unchoking**
  - The client uses a part of its available bandwidth for sending data to random peers
  - The motivation for this mechanism is to avoid bootstrapping problem with the tit for tat selection process and ensure that new peers can join the swarm

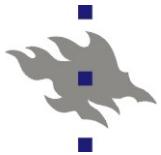


Characteristics of the BitTorrent protocol II

- **Piece selection** is about supporting high piece diversity
  - Local Rarest First for piece selection (start with random, then finally use end game mode)
  - BITFIELD message after handshake with a peer, then HAVE messages for downloaded pieces
- **End game mode**
  - To avoid delays in obtaining the last blocks the protocol requests the last blocks from all peers
  - Sends cancel messages for downloaded blocks to avoid unnecessary transmissions
  - When to start the end game mode is not detailed in the specification

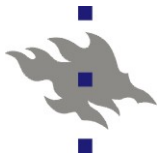
Tit-for-tat in Bittorrent

- Tit-for-tat is a an effective strategy in game theory
  - Idea: cooperate first, and then respond in kind
- Peer has limited number of upload slots
- Upload bandwidth is exchanged for download bandwidth
- If peer is not uploading (only downloading) --> choke
- Upload slot to a random peer (optimistic unchoke)
- Searches for cooperative peers



TFT in more detail

1. Sort peers by incoming data rate
2. Reciprocate with top k, k is proportional to the square root of the upload capacity
3. Optimistically unchoke one other peer
4. Send each peer selected an equal split of capacity



Data transport in BitTorrent

Typically, BitTorrent uses **TCP** as its transport protocol for exchanging pieces, and it uses HTTP for tracker comms.

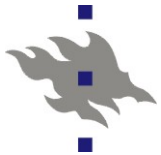
Possible to use HTTP port and real/fake HTTP headers for transport to avoid throttling (not in the specification)

The well known TCP port for BitTorrent traffic is 6881-6889 (and 6969 for the tracker port).

The DHT extension (peer-to-peer tracker) uses various UDP ports negotiated by the peers.

Web seeding (extension)
Use HTTP to download pieces from Web sites

Security extensions (similar to TLS: message stream encryption)



NAT traversal

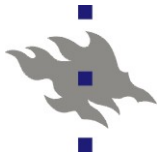
Open ports in firewall/NAT device

UPnP configuration

SSH tunnelling

HTTP tunnelling/proxying

- Any traffic through NATs
- Not necessarily efficient (with relay)



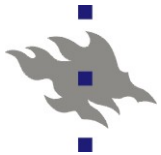
Distributed Tracker

BitTorrent Mainline DHT

Based on Kademlia DHT

Find peers through the DHT network

We will examine Kademlia later on this course

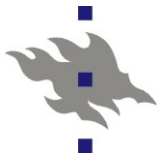


Altruism in BitTorrent

Seeders keep file available

A peer can choose to stay in the network and become a seeder, or leave

Upload activity is also example of altruistic behaviour



Biased neighbor selection

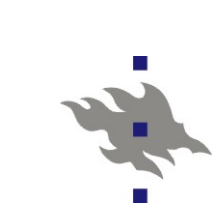
A technique called **biased neighbor selection** has been proposed for reducing cross-ISP traffic

A BitTorrent peer chooses most of its neighbors from the local ISP, and only a few peers from other ISPs.

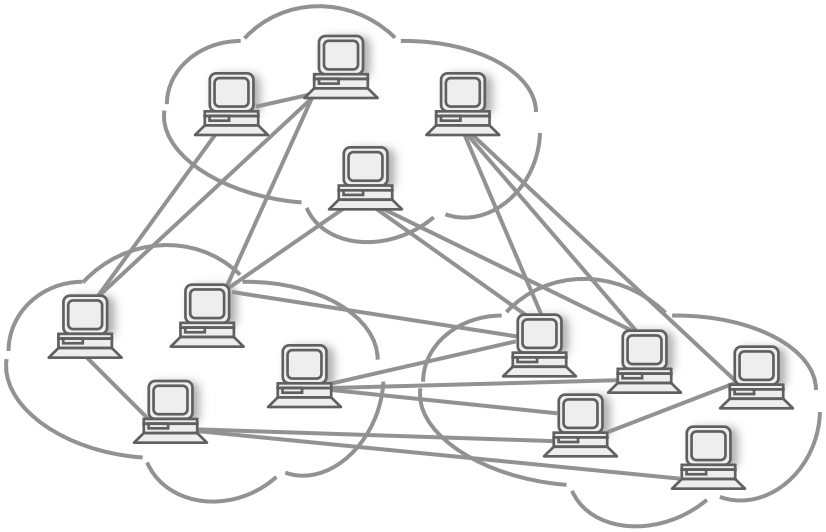
Essentially, the peer selection is biased towards local peers. A parameter *k* represents the number of external peers from other ISPs. The tracker is modified to select 35 − *k* internal peers and *k* external peers that are returned to the client requesting a peer list for a torrent.

If there are less than 35 − *k* internal peers, the client is notified by the tracker to try again later.

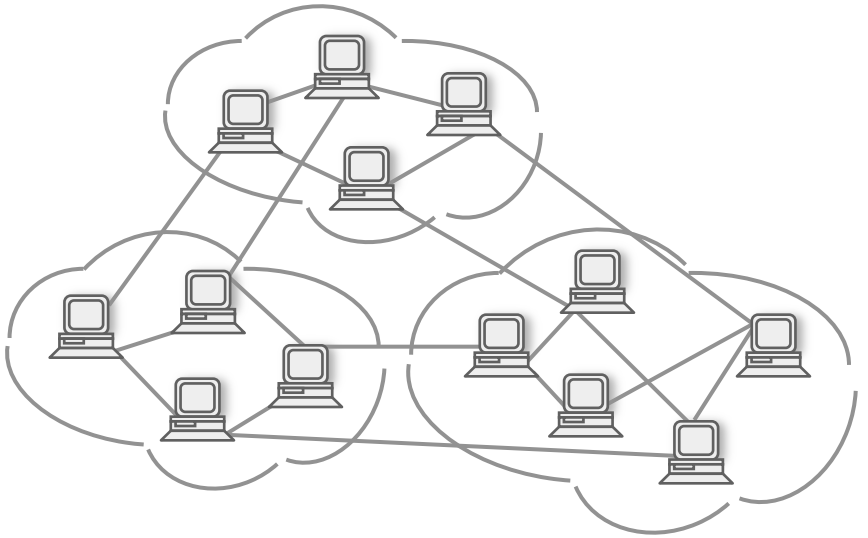
The biased neighbour selection technique works well with the rarest first replication algorithm of BitTorrent; however, other piece selection algorithms, such as random selection, may not lead to optimal performance



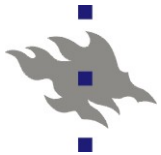
BitTorrent: Effects of Network Topology



Uniform random neighbor selection



Biased neighbor selection

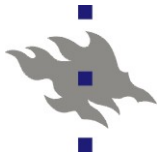


Modelling BitTorrent

BitTorrent performance has been analyzed in the literature using analytical models, including stochastic and fluid models, extensive simulation experiments, experiments on distributed testbeds (PlanetLab), and by obtaining traces from real clients

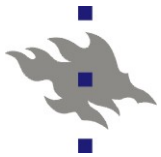
Both analytical and empirical evaluation and estimation are needed to dimension deployments to meet the service capacity demands

Fluid models can be used to analytically estimate the protocol performance and understand the time evolution of the system by using differential equations



Modelling aspects

- Dynamic population model
  - describing the evolution of the peer population in the P2P system
- Peer arrival process
  - steady arrival rate, smoothly attenuating arrival rate, or flash crowd?
- Efficiency of resource sharing
  - utilization of a peer's upload capacity
  - effect of the piece/peer selection policy
  - number of parallel connections
- Selfishness / altruism
  - part of peers are free-riders that do not want to share upload capacity
- Download and upload rates
  - homogeneous or heterogeneous peer population?
- Number of permanent seeds
  - correspond to servers in the client-server architecture



Arrival processes

Various different arrival processes for new peers have been considered in the literature. The three key scenarios are as follows:

- The steady flow scenario used above assumes that new peers appear with a constant rate
- The flash crowd scenario, considers the case where a (large) number of peers appear at the same time after which no new peers arrive
- In a third scenario, the arrival rate is high in the beginning but smoothly attenuates as time passes

Stochastic vs deterministic modelling

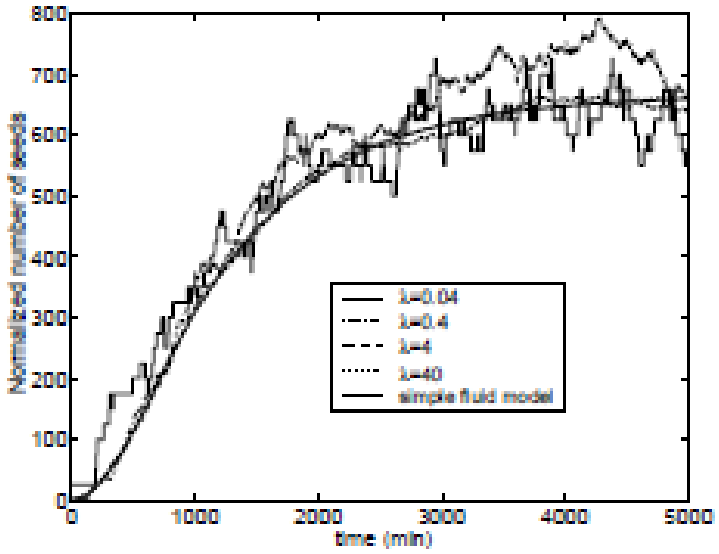


Figure 1: Experiment 1 : The evolution of the number of seeds as a function of time

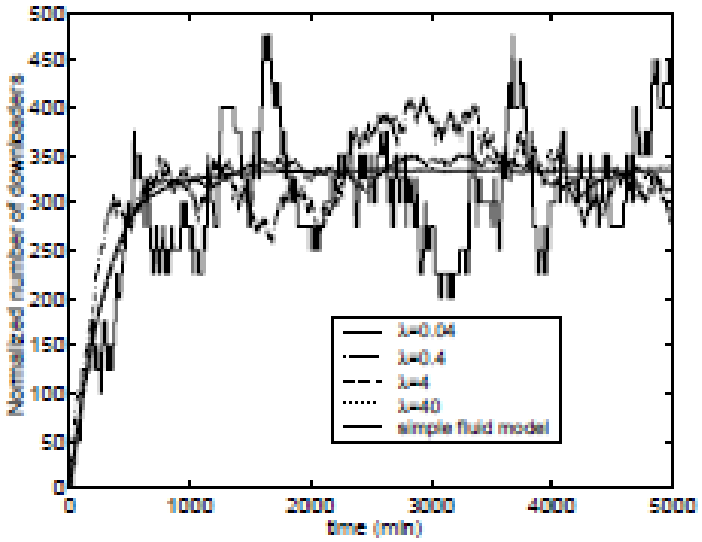
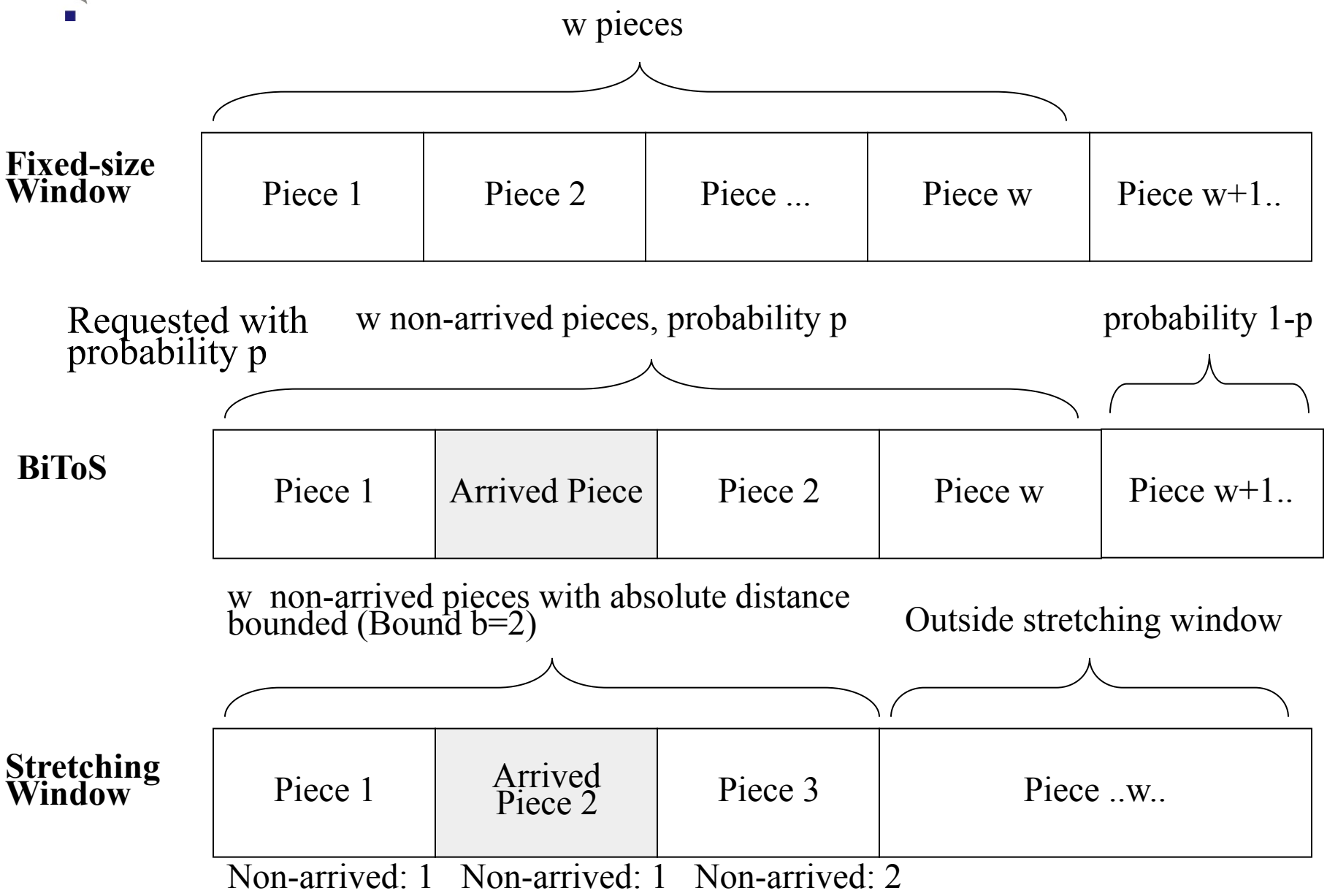


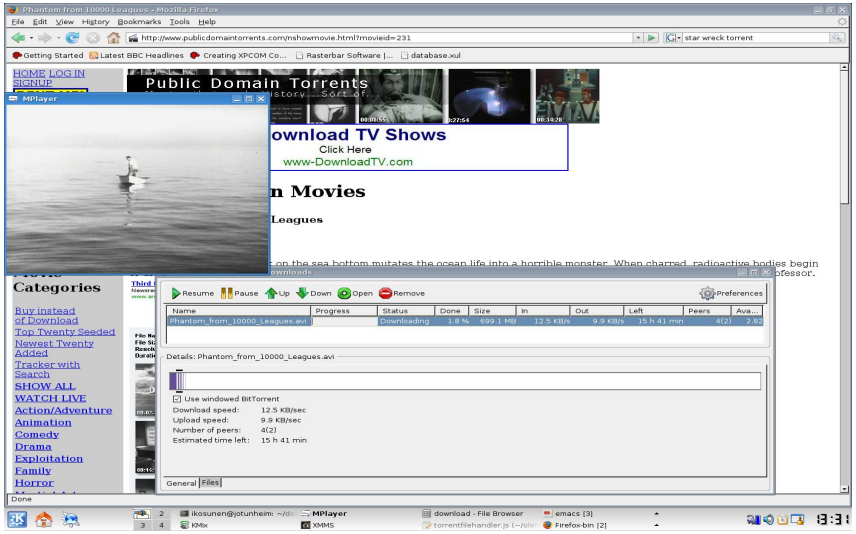
Figure 2: Experiment 1 : The evolution of the number of downloaders as a function of time

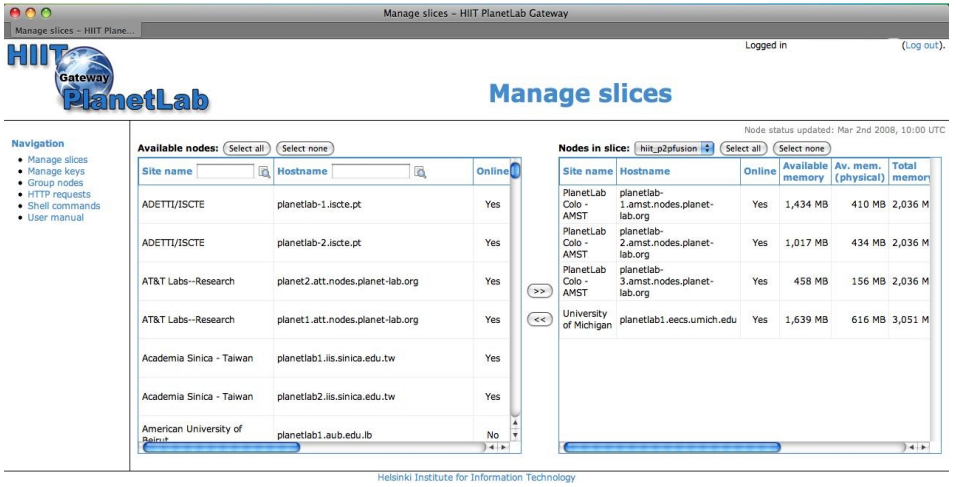
D. Qiu and R. Srikant. Modelling and performance analysis of BitTorrent like peer-to-peer networks. In ACM SIgcomm, pp. 367-378, 2004.

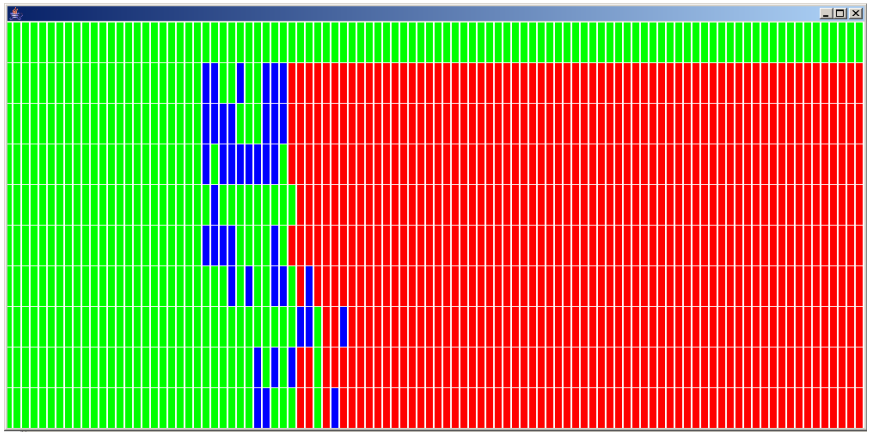
Video-on-Demand

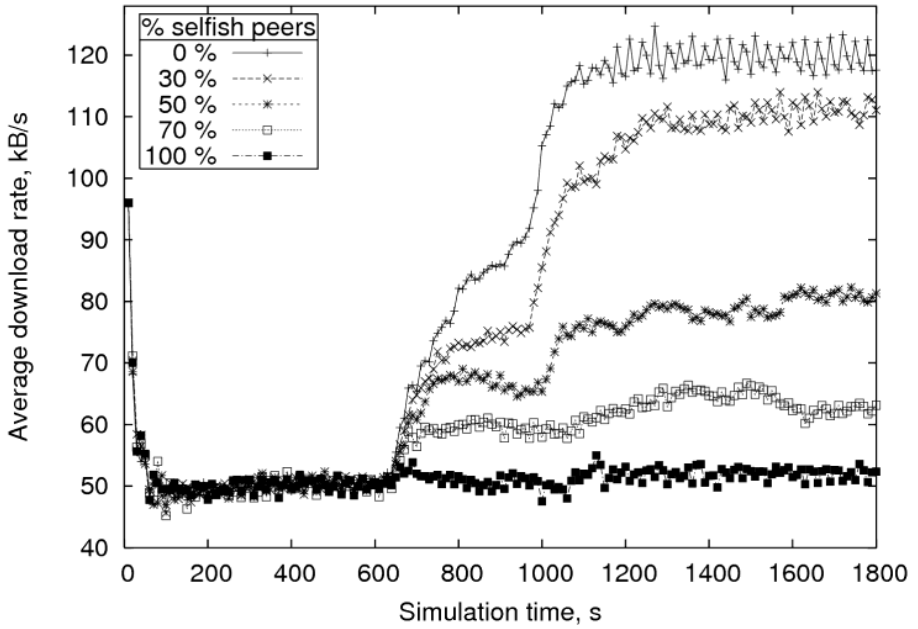


VoD Examples









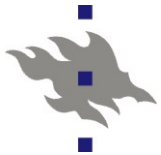
Free-riding and tragedy of the commons

Users of P2P file sharing networks, such as Gnutella, face the question of whether or not to share resources to other peers in the community

They face essentially a social dilemma of balancing between common good and selfish goals

The selfish behaviour often encountered in P2P networks in which peers only download files and do not make resources available on the network is called *free-riding*

Free-riding occurs because the peers have no incentives for uploading files. Free-riding becomes a major problem when significant numbers of peers consume network resources while not contributing to the network. In the context of P2P this is often referred to as *tragedy of the digital commons*

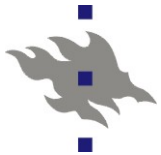


Preventing free-riding

BitTorrent has several mechanisms

- Peer selection: tit-for-tat
- Optimistic unchoking
  - Two uses: find good peers and allow new peers to bootstrap

Other solutions have been proposed as well



BitTyrant (NSDI 2007)

Observation: BitTorrent peers are altruistic

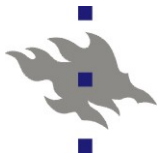
Incentives do not build robustness

A selfish BitTorrent client

Optimize return-on-investment (upload)

- Dynamically set the upload rate to maximize download rate

Can boost download speed by 70%



Building BitTyrant

Key idea: maximize return on investment (RoI)

strategic peer selection

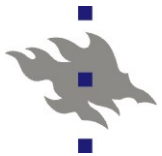
strategic upload rate allocation

Cost: upload rate to peer p, u_p

Benefit: download rate from peer p, d_p

BitTyrant dynamically estimates these rates each tit-for-tat round

www.cs.utexas.edu/~yzhang/Teaching/cs386m-f10/Slides/3-2.**ppt**



Source: www.cs.utexas.edu/~yzhang/Teaching/cs386m-f10/Slides/3-2.**ppt**

Each TFT round, order and reciprocate with peers:

$$\underbrace{\frac{d_0}{u_0}, \frac{d_1}{u_1}, \frac{d_2}{u_2}, \frac{d_3}{u_3}, \frac{d_4}{u_4}}, \cdots$$

choose $k \mid \sum_{i=0}^{k} u_i \leq cap$

After each round, for each peer:

Does not unchoke

If peer reciprocates:

$d_p \leftarrow$ direct observation

...and continues to do so:

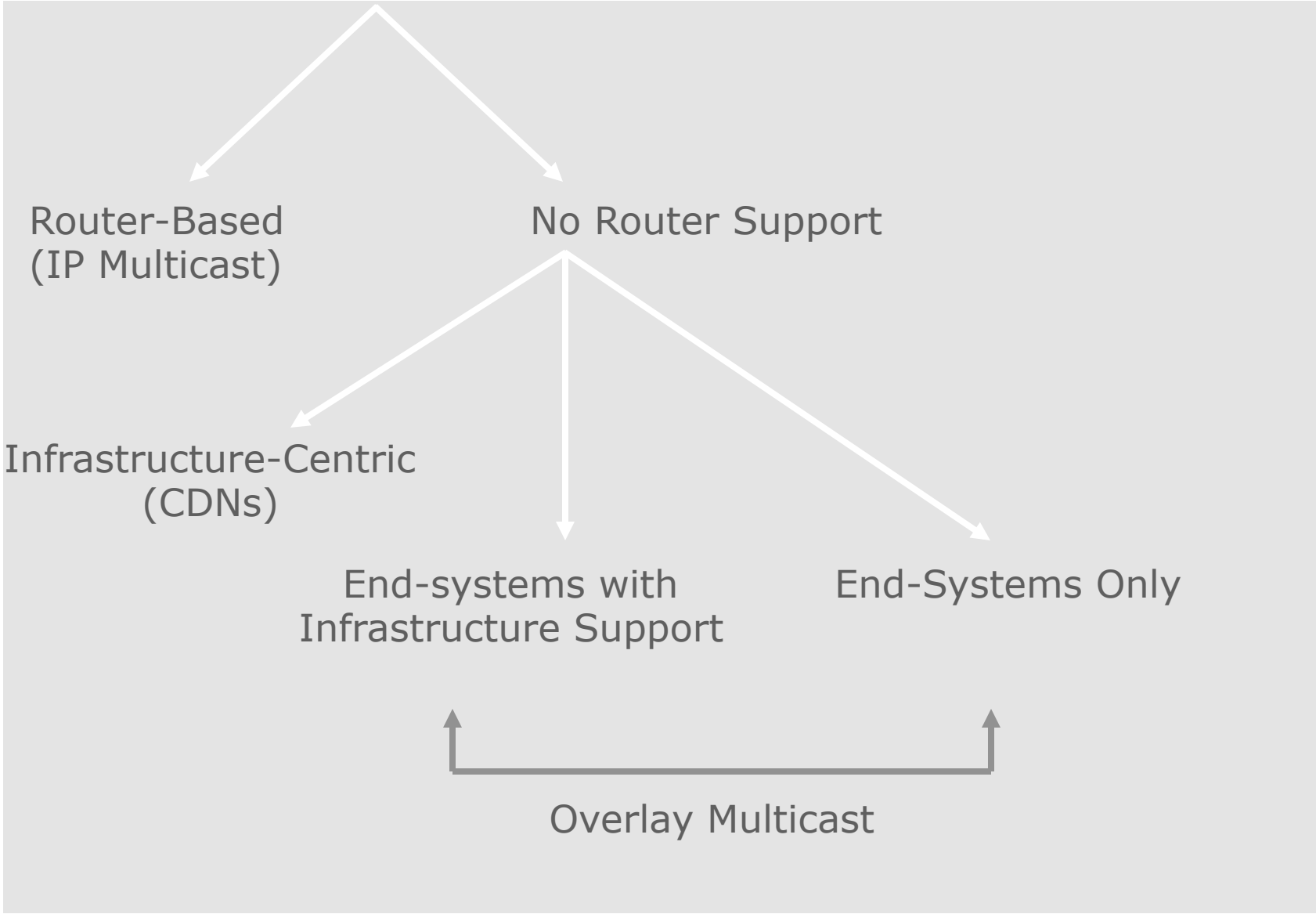
Reduce u_p

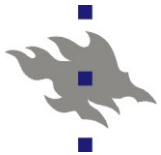
No reciprocation:

Increase u_p

Unchokes

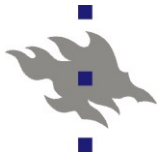
Comparison to IP multicast





Comparison of Multicast Techniques

	IP multicast	**Overlay multicast**
Deployment	Multicast-capable routers	Deployed over the Internet
Multicast structure	Tree, interior nodes are routers, leaves are hosts	Typically a tree, both interior nodes of the structure and leaves are hosts
Transport layer protocol	UDP	TCP or UDP
Scalability	Limited	High (depends on solution)
Congestion control / recovery	No	Various, can utilize unicast (TCP) for node-to-node reliability
Efficiency	High	Low (varies), can suffer from high stretch and unoptimal interdomain routing
Example protocols	Protocol Independent Multicast (PIM), Core-based Trees (CBT), …	BitTorrent variants, Scribe, SplitStream, OverCast, …



	BitTorrent
Decentralization	Centralized model
Foundation	Tracker
Routing function	Tracker
Routing performance	Guarantee to locate data, good performance for popular data
Routing state	Constant, choking may occur
Reliability	Tracker keeps track of the peers and pieces